前坂 誠二 ／ 大橋 拓也

株式会社ミガロ.

システム事業部 システム2課

[Delphi/400] iOSモバイルアプリ開発のデザイニングテクニック

略歴　前坂 誠二
1989 年 3 月 21 日生まれ
2011 年 関西大学文学部卒業
2011 年 4 月 株式会社ミガロ. 入社
2011 年 4 月 システム事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業を担当。Delphi、Delphi/400 の開発経験を積みながら、日々スキルを磨いている。

略歴　大橋 拓也
1992 年 7 月 4 日生まれ
2015 年 龍谷大学経営学部卒業
2015 年 4 月 株式会社ミガロ. 入社
2015 年 4 月 システム事業部配属

現在の仕事内容
Delphi、Delphi/400 を利用したシステム開発および保守作業を担当。開発スキルの向上を目指し、日々精進している。

1.はじめに

ここ数年で、企業でのスマートデバイス普及率は大きく跳ね上がり、それに伴ってモバイルアプリケーション開発の需要も高まっている。

モバイルアプリケーションの開発にあたり、多くの開発者が向き合うことになるのが画面デザインの部分ではないだろうか。なぜなら、PC アプリケーションとモバイルアプリケーションでは使用できる画面領域や操作方法が違うため、モバイルアプリケーション用に再デザインする必要があるからだ。

そこで本稿では、モバイルアプリケーション開発で課題となる画面デザインについて考察し、それらの課題を解消する開発テクニックを解説する。

スマートデバイスにはさまざまな種類があるが、企業で業務用に導入されているデバイスで圧倒的に多いのが iOS の iPad である。そのため、本稿ではデバイスとして iPad（mini）を題材に、Delphi/400 XE7 を使用する。もちろん画面デザインの考え方は、他のデバイスや OS であっても同じである。

2.iOSアプリケーションで使用できる画面領域

まず、PC（Windows）と iOS デバイスで作成されるアプリケーションの画面サイズについて整理する。

PC アプリケーションでは、端末のサイズにもよるが、1280 × 1024 や 1366 × 768 を最大サイズとして作成されることが多い。

それに対して、iOS デバイスのアプリケーションでは iPad で 768 × 1024、iPhone で 320 × 568 や 414 × 736 が最大サイズである。こうして見ると、意外にも iPad についてはサイズに大きな違いはないことがわかる。【図 1】

しかし PC と iOS デバイスのアプリケーションでは、操作方法が大きく異なる。PC では主にマウスやキーボードでの操作が多いので、画面項目のサイズや項目間の余白についてはあまり気にする必要はない。

それに対して iOS デバイスでは、主に指によるタッチ操作が主流となる。項目間の余白が少なかったり、項目自体のサイズが小さいと、ボタンの押し間違いや項目の選択ミスにつながる。

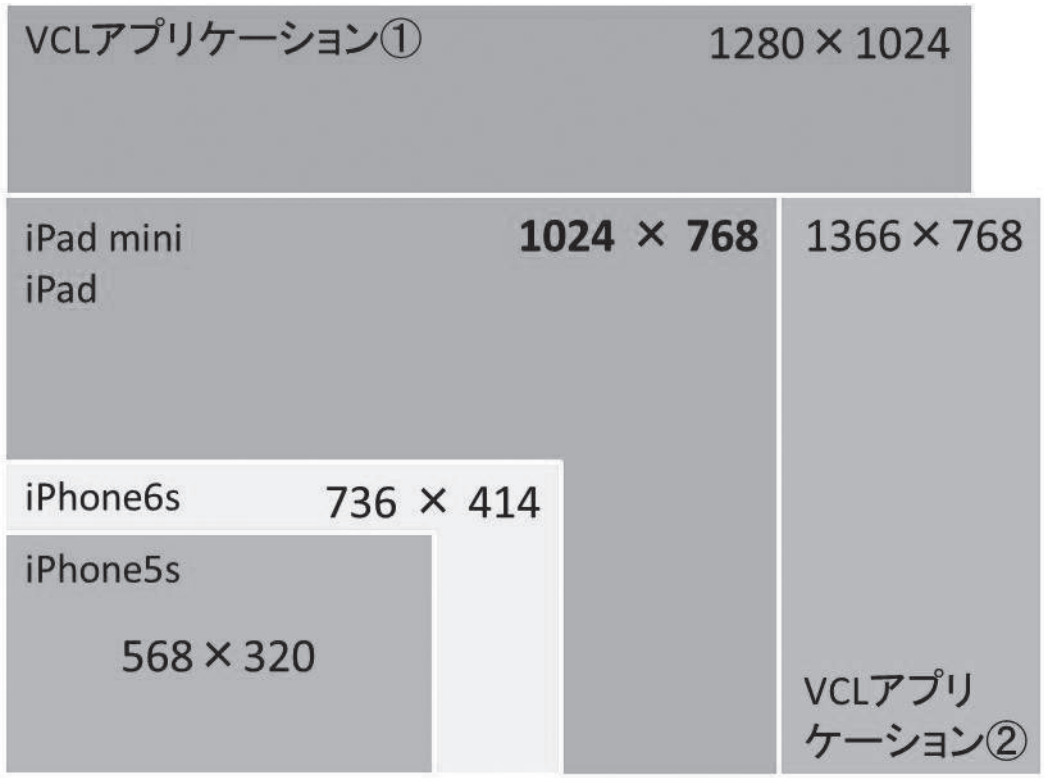
さらに iOS デバイスではキーボードが画面下から表示されるため、画面下部の領域は使用できなくなる可能性もある。

つまり iOS アプリケーションでは、画面サイズ自体の大きさは似ていても、項目間の余白や項目自体のサイズを意識した画面デザインを行うと、実際に使用できる画面領域は PC よりもかなり小さくなる。

3.画面デザインのポイント

ここまで、PC と iOS デバイスにおける画面サイズの違い、iOS デバイスで使用できる画面領域が少なくなる要因について解説した。次はそれらの内容を踏まえ、iOS モバイルアプリケーションの画

図1



※iOSはポイントで比較

図2

面デザインのポイントについて説明する。

1つ目のポイントは、コンポーネントの配置や表示方法を考慮することである。たとえばボタンであれば、アイコンのような見た目で表示することで余分なスペースを取らずに使用できる。またキーボードが表示された際には、画面全体をスクロール表示させることにより、見えなくなった領域も表示可能となる。

2つ目のポイントは、一度に表示する画面項目をなるべく少なくすることである。そのためにはスライドやポップアップを使用し、必要な場合にのみ項目を表示させる方法が有効である。

ただしこのとき、注意すべき点がある。それは、ポップアップやスワイプが有効となる条件をアプリケーション全体でルール決めしておくことである。ポップアップやスワイプが有効となる条件が各画面によってバラバラだと、使用するユーザーが混乱するからだ。

つまり、画面デザインのポイントは大きく以下の2点となる。

①コンポーネントの配置や表示方法を考慮する（4.で詳しく解説）
②必要な場合にのみ必要な情報を表示させる（5.で詳しく解説）

これらのポイントを考慮して開発することで、操作性を損なうことなく画面領域を有効活用できる。

4.画面領域を考慮したデザイン実装例①

前述した画面デザインのポイントを踏まえ、iOSモバイルアプリケーションでの実装方法について順に説明する。

なお、本稿の実装例ではVCLで作成したPCアプリケーションを元に、iOSアプリケーションを作成することを想定している。各機能の全体図は、【図2】のとおりである。

4-1. 新規プロジェクトの作成

まずは、[ファイル|新規作成]より「マルチデバイスアプリケーション-Delphi」を選択する。選択時にダイアログでテンプレート選択画面が表示されるので、「空のアプリケーション」を選択する。これでモバイルアプリケーションのプロジェクトが新規作成できる。【図3】

iOSモバイルアプリケーションの開発には、FireMonkeyを用いる。実装例の解説に入る前に、VCLでの作成時と異なる点について少し触れたい。

まず、「コンポーネントの階層化」についてである。VCLでは、TPanel等のコンテナコンポーネント群やフォームについては親コンポーネントとすることができる。それに対し、FireMonkeyではコンテナコンポーネントやフォームのみに制約されず、他のコンポーネントでも自由に親子関係をもたせることができる。【図4】

こうした階層化により、親コンポーネントに対する処理を子コンポーネントにも反映できる。またそれ以外にも、デザイン時にはオブジェクトをまとめて移動できるといった利点もある。

次に、「Visibleプロパティ」について説明する。Visibleプロパティは、項目の表示・非表示を設定するプロパティである。

VCLでは、設計画面上で設定値（True/False）を切り替えたとしても、常に表示された状態である。それに対してFireMonkeyの設計画面では、Visibleプロパティを Falseとした場合、設計画面上でも非表示となる。

そのため、非表示にしているコンポーネントを編集する際は、一時的にVisibleプロパティをTrueにして編集する必要がある。【図5】

その他にも細かい違いが多くあるが、本稿の実装例では以上の違いを念頭に置けば問題ない。

4-2. ポイント①の実装例

まずは、ポイント①「コンポーネントの配置や表示方法」を考慮した画面デザインの実装方法を説明する。こちらの実装例は、以下のとおりである。

・ボタンのアイコン化
・画面全体に対するスクロールの実装

4-3. ボタンのアイコン化の実装

PCアプリケーションでは、キャプションを表示してボタンの意味を伝えるのが一般的である。本稿の例でも、PCアプリケーションではボタンに「検索」とキャプションを設定している。

それに対してiOSモバイルアプリケーションでは、アイコン化することで画面領域を有効的に使用できる【図6】。実装方法も非常に容易なので、限られた画面領域を有効活用する最もシンプルなテクニックであるといえる。

実装方法は、[ツールパレット|Standard]よりTButtonを選択し、設置する。設置後、StyleLookUpプロパティから［searchtoolbutton］を選択する【図7】。以上の操作で、簡単にボタンをアイコン化できる。アイコンの見た目は、デバイスのOSによって用意されている。

しかし、このままではアイコンに枠や背景色がないので、タップできるかどうかの判断が難しい。そこで、TRectAngleコンポーネントと組み合わせる。これにより、枠や背景色を指定してカスタマイズできる。【図8】

さらに［ツールパレット|Shapes］からTRectAngleを選択する。設置後、[構造ビュー］にて設置したRectAngle1にButton1をドラッグ＆ドロップすることで、コンポーネントを階層化させる。【図9】

あとは、RectAngle1のColorプロパティやX/YRadiusプロパティ（オブジェクトの角の丸さを調整するプロパティ）を任意に変更することで、自由に枠や背景色を指定できる。実装例での設定プロパティは、【図10】のとおりである。

またPCのマウス操作と違い、スマートデバイスのタッチ操作では、意図したタッチポイントとずれて誤動作となることも多い。そのため、TRectAngleの上に配置しているTButtonコンポーネントは、TRectAngleよりもサイズを大きく設定しておいたほうが、タッチに反応しやすく、操作性がよくなる。

4-4. スクロール機能の実装

スクロール機能の実装は、入力項目の多い画面に有効なテクニックである。iOSデバイスでは画面項目への入力時に、画面下部にキーボードが表示される。その際、入力項目部全体にスクロールを設定しておくことで、項目がキーボードに隠れても自由にスクロールできる。

実装方法としては、［ツールパレット|Layouts］よりTVertScrollBoxを選択する。入力項目部に設置し、Alignプロパティを Clientにする。【図11】

図3

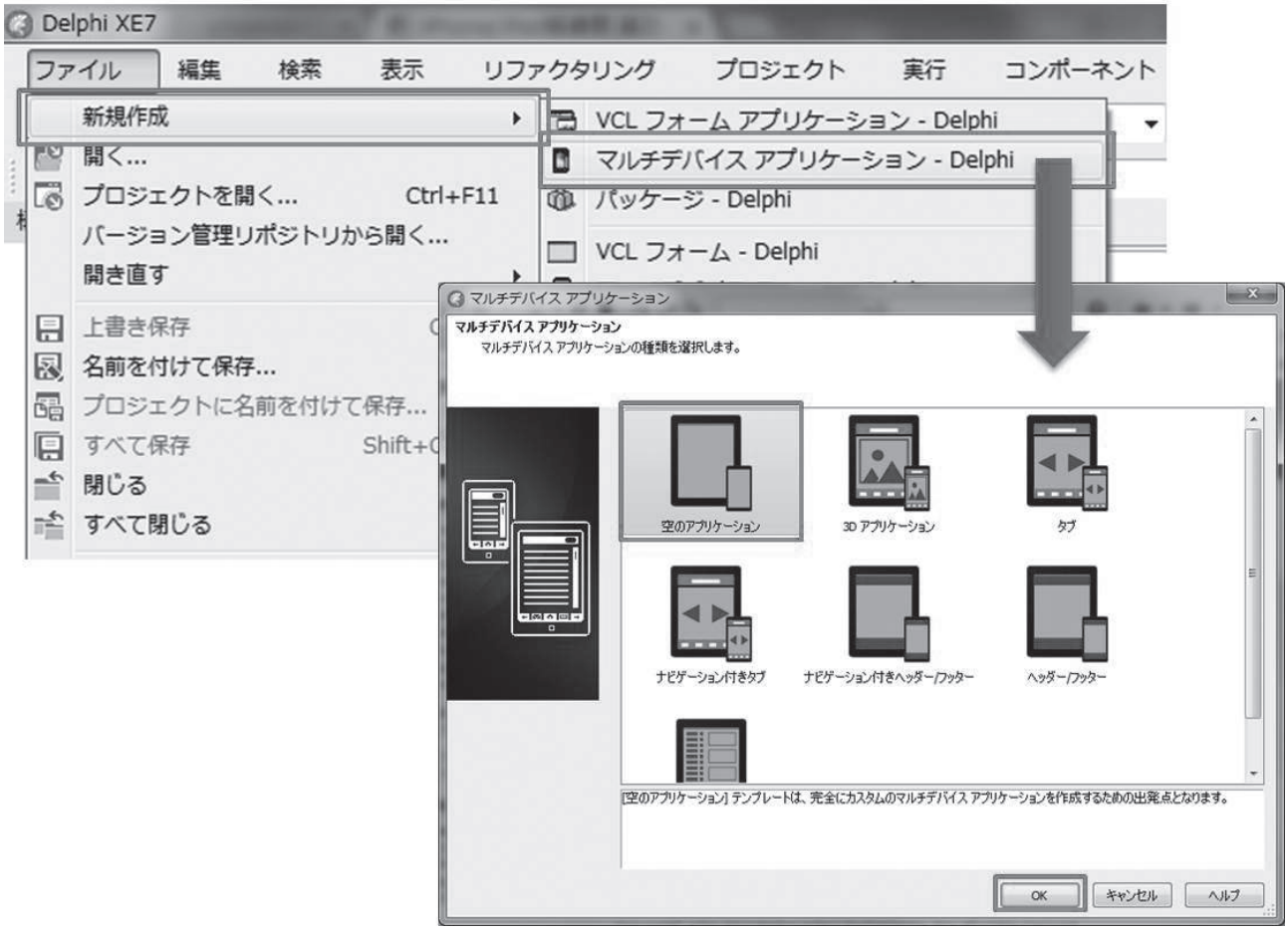

Delphi XE7
ファイル
編集
検索
表示
リファクタリング
プロジェクト
実行
コンポーネント
新規作成
開く...
プロジェクトを開く... Ctrl+F11
バージョン管理リポジトリから開く...
開き直す
上書き保存
名前を付けて保存...
プロジェクトに名前を付けて保存...
すべて保存
閉じる
すべて閉じる
VCL フォーム アプリケーション - Delphi
マルチデバイス アプリケーション - Delphi
パッケージ - Delphi
VCL フォーム - Delphi
マルチデバイス アプリケーション
マルチデバイス アプリケーション
マルチデバイス アプリケーションの種類を選択します。
空のアプリケーション
3D アプリケーション
タブ
ナビゲーション付きタブ
ナビゲーション付きヘッダー/フッター
ヘッダー/フッター
[空のアプリケーション] テンプレートは、完全にカスタムのマルチデバイス アプリケーションを作成するための出発点となります。
OK
キャンセル
ヘルプ

図4


<FireMonkey>
<VCL>
構造
FireMonkeyForm
Edit1
Button1
構造
VCLForm
Edit1
Button1
VCLでは階層化できないコンポーネント同士でも階層化が可能

あとは、任意でVertScrollBox1上に画面項目を設置していく。これにより入力項目部全体を自由に上下スクロールできるようになり、キーボード表示時に項目が隠れた場合も、下にスクロールすることで表示できる。【図12】

5.画面領域を考慮したデザイン実装例②

5-1. ポイント②の実装

ここでは、ポイント②「必要な場合にのみ必要な情報を表示」を踏まえた画面デザインの実装方法について説明する。本稿では、以下の2点を実装する。

・画面をスワイプすることで側面から情報を表示（スライド機能）
・ボタンタップで情報表示（ポップアップ機能）

こうしたスライド機能やポップアップ機能は実装が難しい、と思われるかもしれない。しかし、FireMonkeyではプロパティの設定や少々のコーディングで容易に実装できる。

5-2. スライド機能の実装

スライド機能は、画面の側面から指をスワイプさせることで隠れたリスト等を表示させる機能である。メニューを表示させたり、選択項目の詳細情報を表示させるなど、用途はさまざまである。

実装例では、PCアプリケーションでフッター部に配置しているボタンをスライド部に実装する。【図13】

［ツールパレット|Common Controls］からTMultiViewコンポーネントを選択し、設置する。その後、設置されたMultiView1のModeプロパティをDrawerに変更する。プロパティを変更するとVisibleプロパティがFalseとなるため、スライド内容編集時はVisibleプロパティを一時的にTrueにする。【図14】

以上で、スライド機能を実装できた。あとは任意にスライドで表示する内容を、【図15】のように設定すればよい。

5-3. ポップアップ機能の実装

ポップアップ機能は、ボタンタップなどの動作で画面項目の表示・非表示を切り替えるテクニックである。

こうしたポップアップ機能は、TPopupというコンポーネントを使用することで実装できる。実装例では、PCアプリケーションでヘッダー部のログイン情報を、ボタンタップで表示できるように実装する。

まず、ログイン情報を呼び出すボタンを設置する（ボタンの実装方法は4-4を参照）。実装例での設定プロパティは、【図16】のとおりである。

次に、ポップアップ部を実装する。まずは、［ツールパレット|Standard］よりTPopupを選択し、表示させたい位置に配置する。【図17】

配置時はVisibleプロパティがFalseとなっているので、編集時はTrueに変更する。続いて、［ツールパレット|Shapes］よりTCalloutRectangleを選択し、プロパティの設定を行う。【図18】

そして、最後に表示項目を配置する。本稿では、ログイン情報とログアウトボタンを実装するため、［ツールパレット|Standard］よりTLabelを必要数設置し、4.で解説したTRectAngleを組み合わせたボタンを配置する。あとは配置したコンポーネントを、【図19】のように階層化すれば完成である。【図20】

なお、ポップアップ表示・非表示の処理に関しては、ソースコードを記載する必要がある。こちらの実装方法については、【ソース1】に示す。

6.エフェクトを用いた画面の実装例

6-1. エフェクトの活用

ここまで画面領域を考慮した開発テクニックを題材に説明してきた。モバイルアプリケーションの開発では、画面領域の考慮のほかに、ユーザーが操作しやすい画面設計を行うことも大切な要素の1つである。

たとえば選択形式の画面項目の場合には、チェックボックスを使用することが多い。もちろん標準のTCheckBoxを使用しても問題はないが、本稿ではTCheckBoxの代わりに、エフェクトコンポーネントを利用した選択用ボタンの作成を推奨する。【図21】

選択用ボタンとは、【図21】のようにボタンタップによって凹凸が変化する機能を実装したボタンであり、以下にその実装方法について記載する。

まずエフェクトコンポーネントを使用する際は、4.で触れたコンポーネントの階層化の考え方が基本となる。この実装例では、TRectAngleにTBevelEffect、TInnerGlowEffect、TLabelを階層化させて、実装する。

6-2. エフェクト機能の実装

まず、TRectAngleとTLabelを画面に設置する。そこへ［ツールパレット|Effects］より非表示コンポーネントであるTBevelEffectおよびTInnerGlowEffectを設置し、各コンポーネントを【図22】のように階層化する。

TBevelEffectでは親コンポーネントの奥行きを設定でき、TInnerGlowEffectは親コンポーネントを内側に向けて発光させる効果がある。

まず、先ほど配置したBevelEffect1とInnerGlowEffect1のプロパティ設定を、【図23】のように設定する。あとは、Label1のOnMouseDownイベントに、【ソース2】のようにコーディングするだけで、実装は完了である。

7.まとめ

本稿では、モバイルアプリケーションの開発の際に課題となることが多い画面デザインに対する工夫を考察し、その具体的なテクニックを説明してきた。

iOSモバイルアプリケーションでは、配置する項目のサイズや項目間の余白を意識して画面デザインする必要があるので、使用できる画面領域は必然的に狭くなる。

本稿で紹介した画面領域を考慮した開発テクニックは、すべて容易に実装でき、初めてモバイルアプリケーションを開発する方であっても十分活用できる。

またモバイルアプリケーションの画面デザインで最も参考になるのは、実際のストアなどに公開されているアプリケーションである。とくにダウンロード数が多いアプリケーションは、利用するユーザーが多い分、画面デザインや操作性も洗練されていることが多い。

モバイルアプリケーションはデザインの工夫次第で、使い勝手が大きく変わるので、優れた画面設計や実装テクニックをどんどん取り入れていくことが重要である。 M

図5

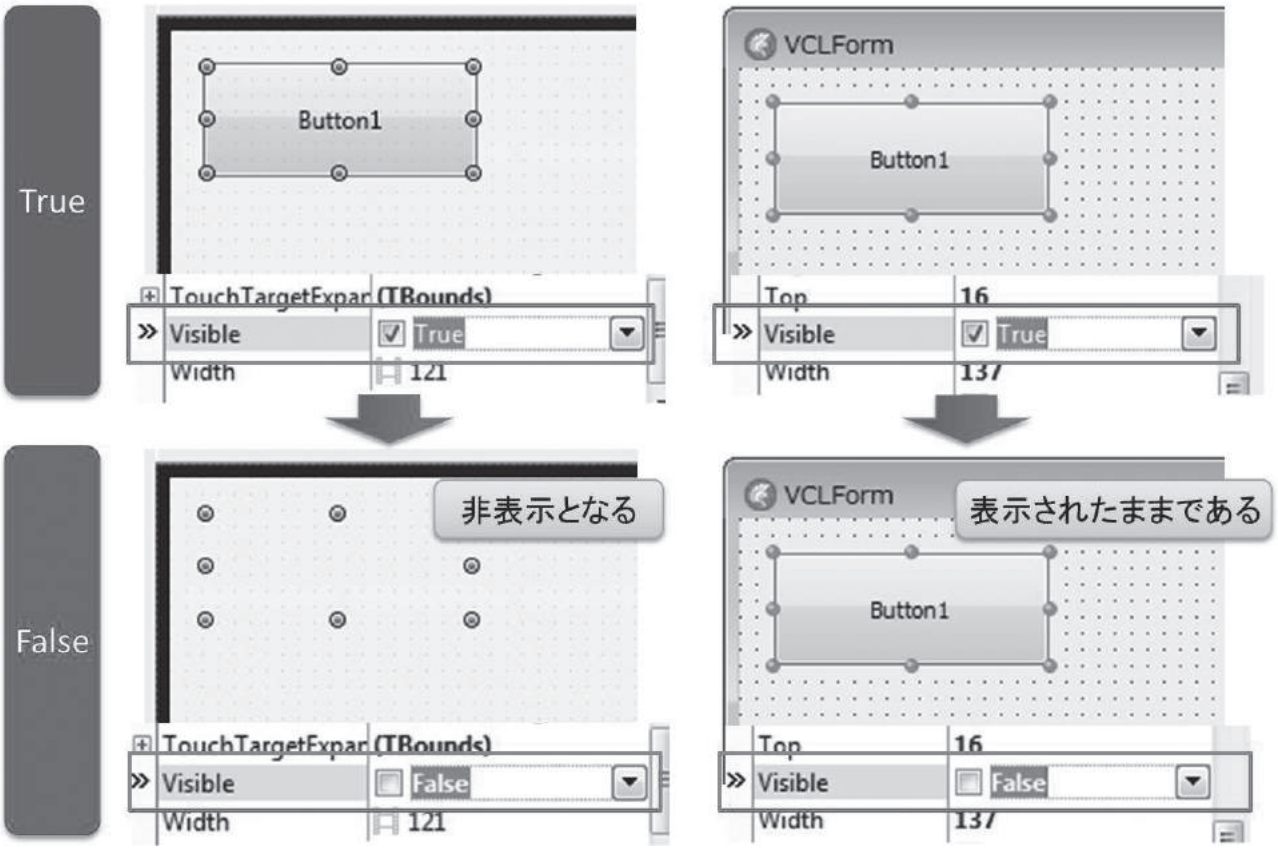

True
Button1
TouchTargetExpan (TBounds)
Visible
True
Width
121
VCLForm
Button1
Top
16
Visible
True
Width
137
False
非表示となる
TouchTargetExpan (TBounds)
Visible
False
Width
121
VCLForm
表示されたままである
Button1
Top
16
Visible
False
Width
137

図6

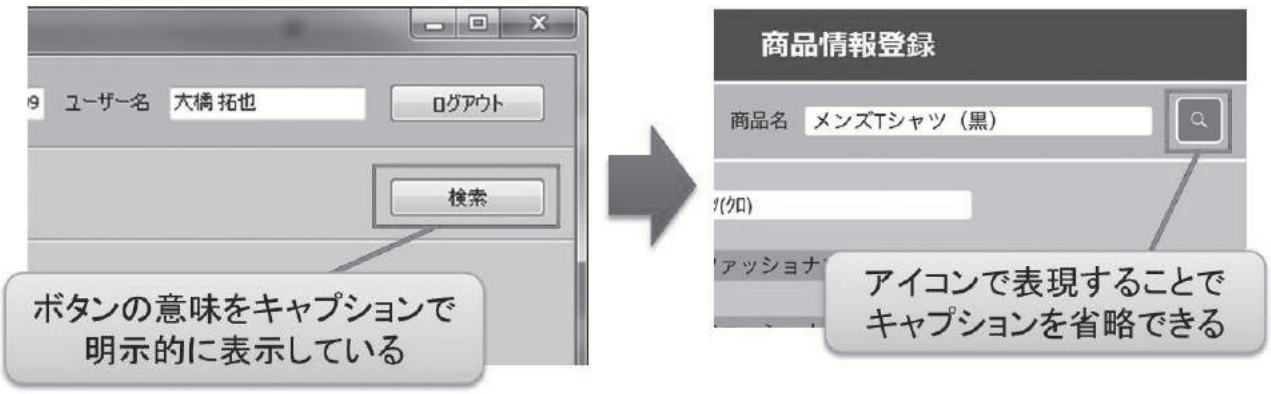
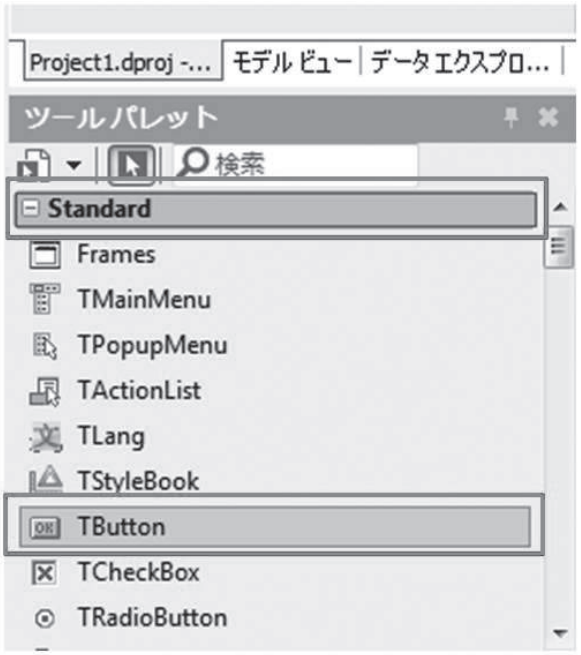
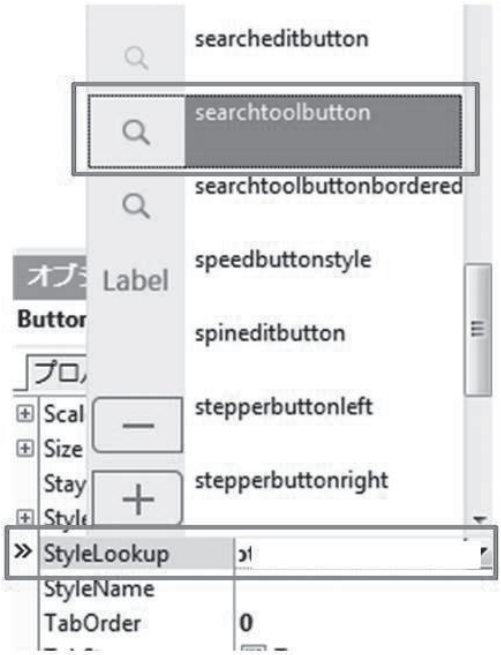

ユーザー名
大橋 拓也
ログアウト
検索
ボタンの意味をキャプションで
明示的に表示している
商品情報登録
商品名
メンズTシャツ（黒）
アイコンで表現することで
キャプションを省略できる

図7

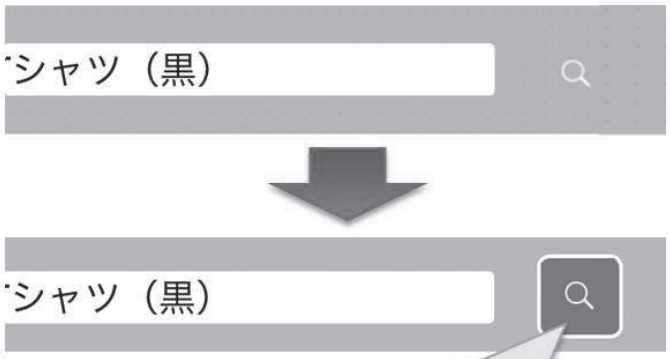


ツールパレットからTButtonを選択



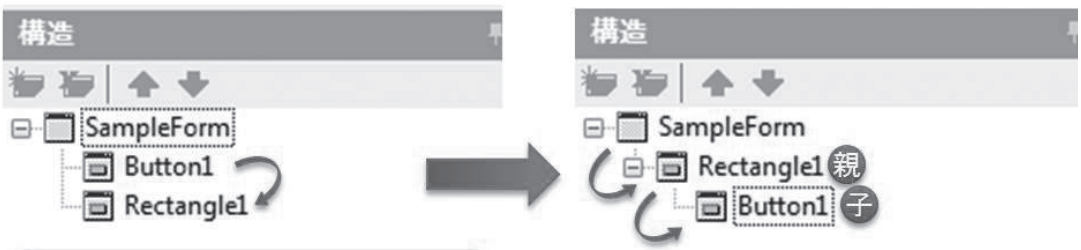
StyleLookupプロパティ設定

図8



TRectAngleと組み合わせることで
枠や背景色を設定することができる

図9



ドラッグ&ドロップで
親子関係を持たせる

階層構造ができる

図10

Button1設定プロパティ

Align	Center
StyleLookup	searchtoolbutton

RectAngle1設定プロパティ

Hight	38
Width	38
XRadius	5
YRadius	5

図11

図12

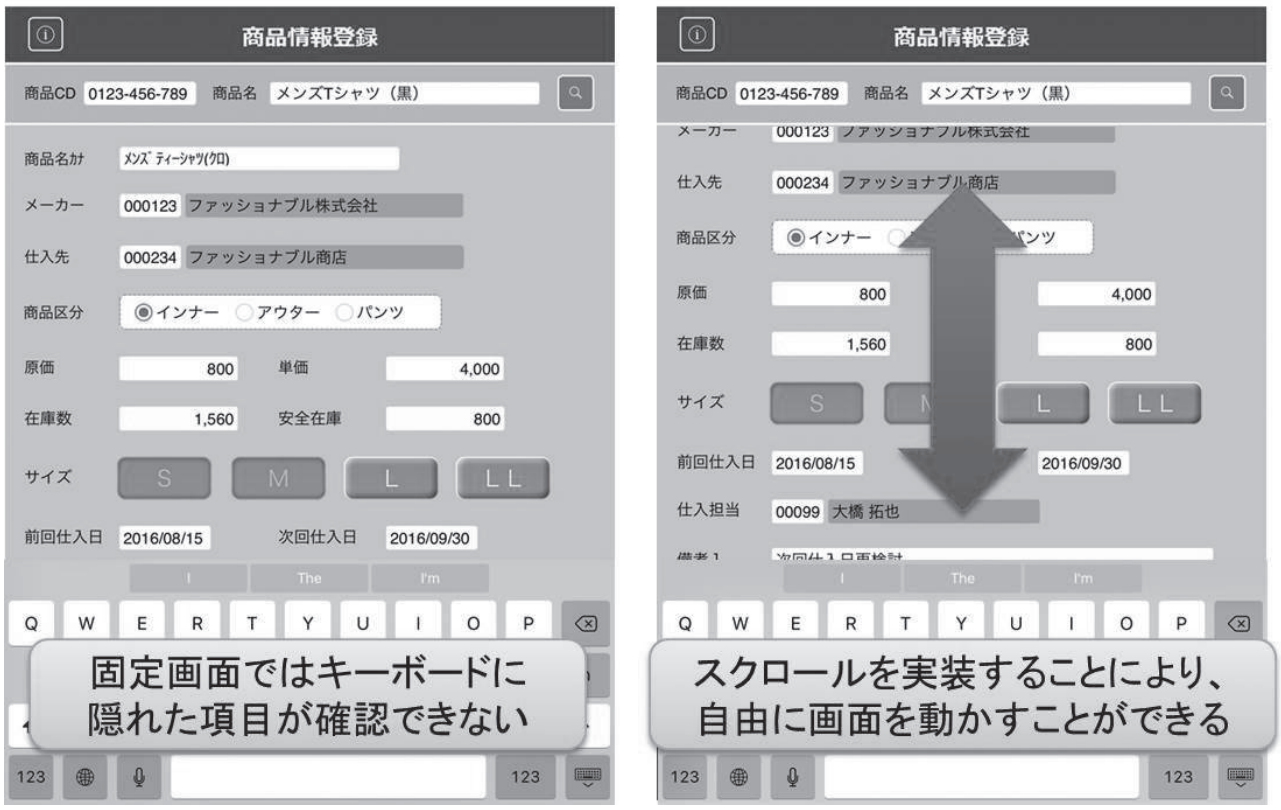

商品情報登録
商品CD 0123-456-789
商品名 メンズTシャツ（黒）
商品名カナ
メーカー 000123 ファッショナブル株式会社
仕入先 000234 ファッショナブル商店
商品区分 インナー アウター パンツ
原価 800
単価 4,000
在庫数 1,560
安全在庫 800
サイズ S M L LL
前回仕入日 2016/08/15
次回仕入日 2016/09/30
固定画面ではキーボードに
隠れた項目が確認できない
仕入担当 00099 大橋 拓也
スクロールを実装することにより、
自由に画面を動かすことができる

図13

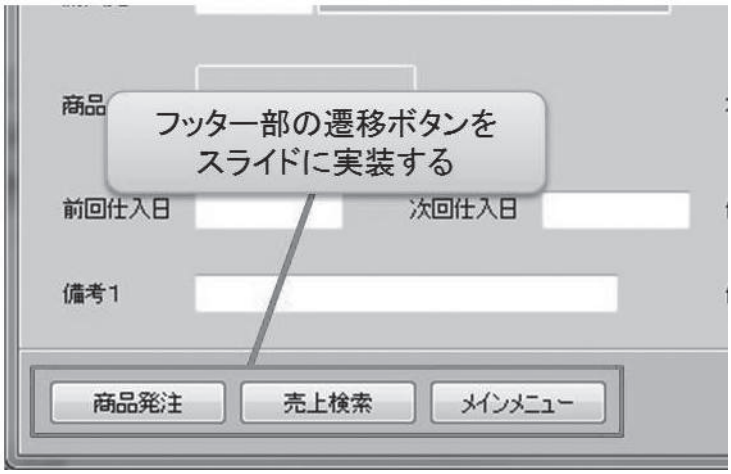

フッター部の遷移ボタンを
スライドに実装する
前回仕入日
次回仕入日
備考1
商品発注
売上検索
メインメニュー

図14

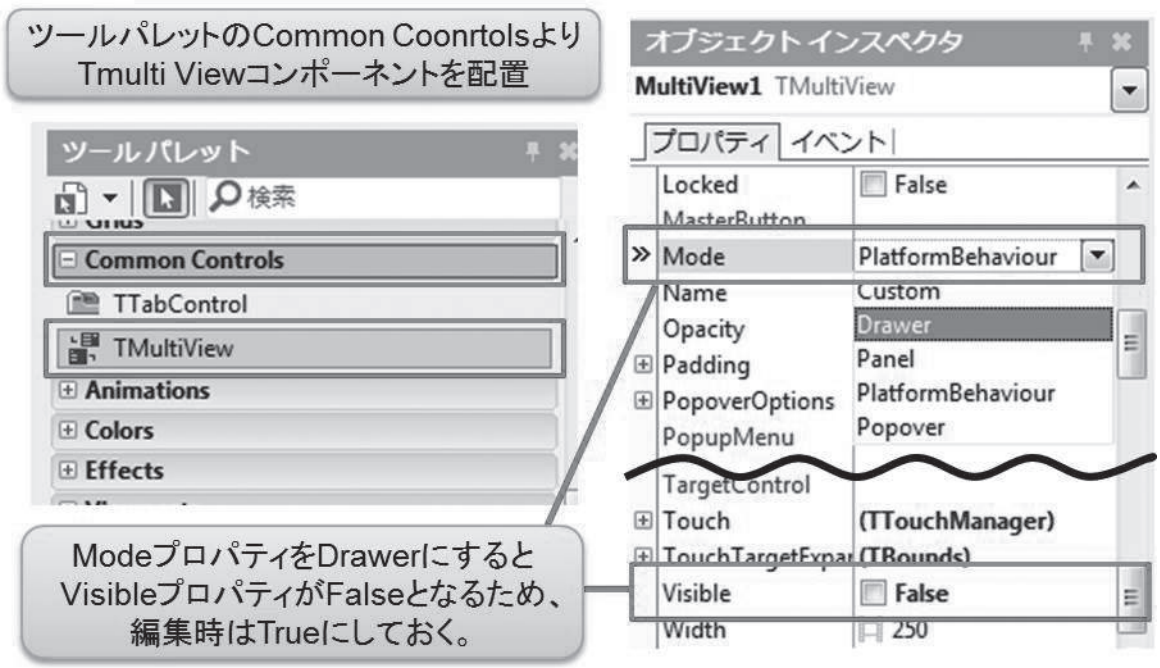


図15

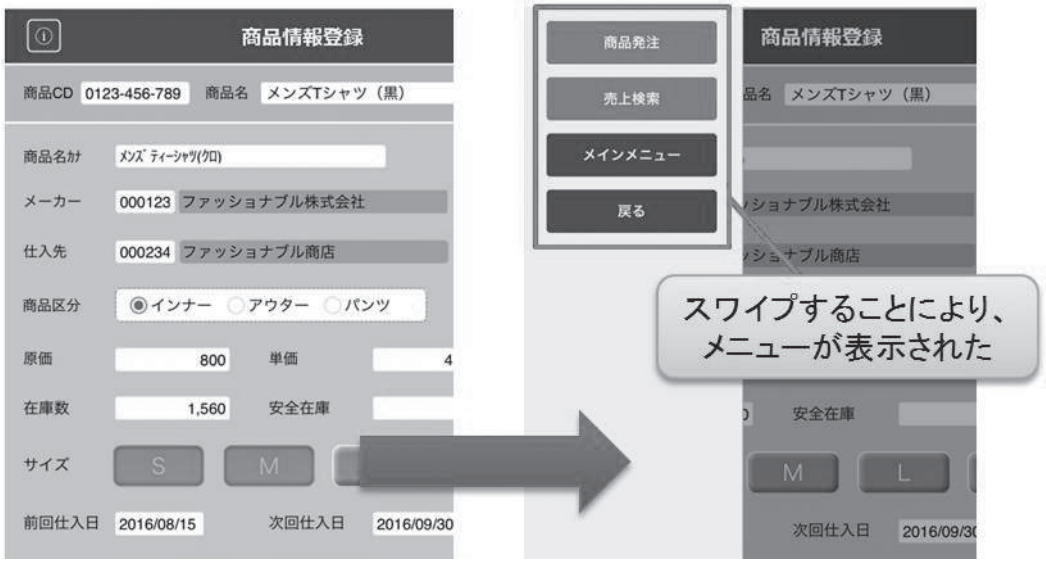


図16

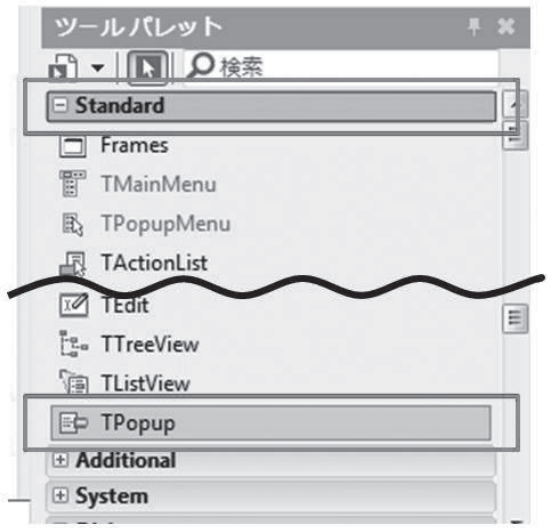
Button2設定プロパティ

Align	Center
StyleLookup	infotoolbutton

RectAngle2設定プロパティ

Fill - Color	Navy
Stroke - Color	White
XRadius	5
YRadius	5

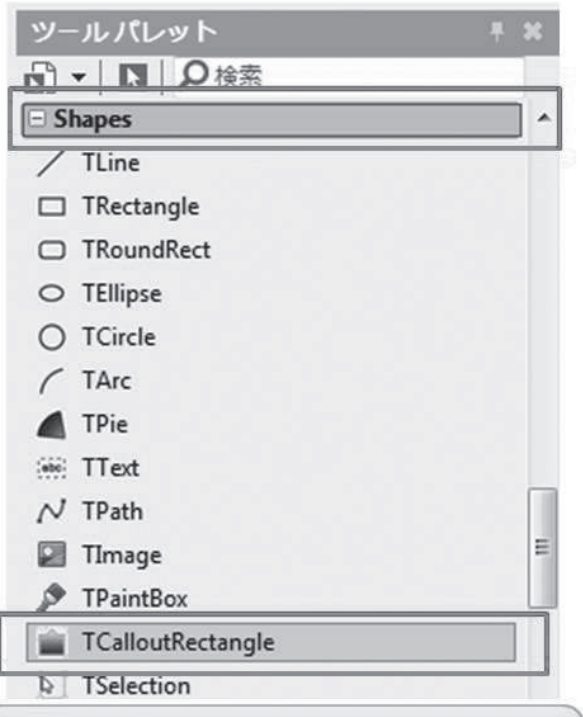
図17



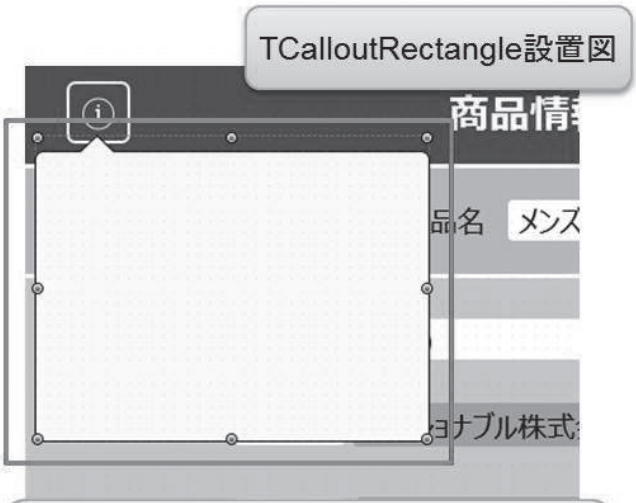
ツールパレットのStandardより
TPopupを設置する

ポップアップ表示をさせたい位置に
配置し、任意のサイズに設定する

図18



ツールパレットのShapesより
TCalloutRectangleを設置する



CalloutOffsetプロパティにより
吹き出し元の位置を変更できる

CalloutRectangle1設定プロパティ

Align	Client
CalloutOffset	31
Fill - Color	White
XRadius	5
YRadius	5

図19

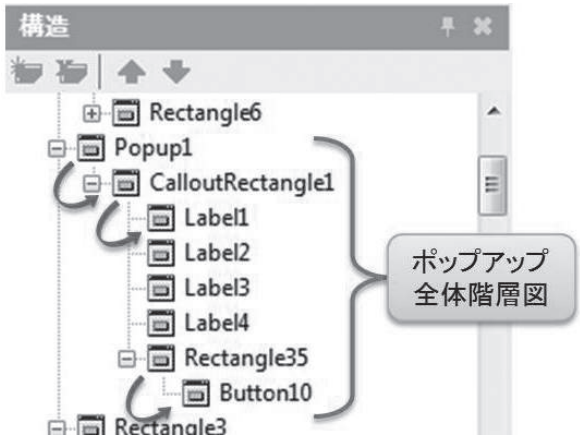
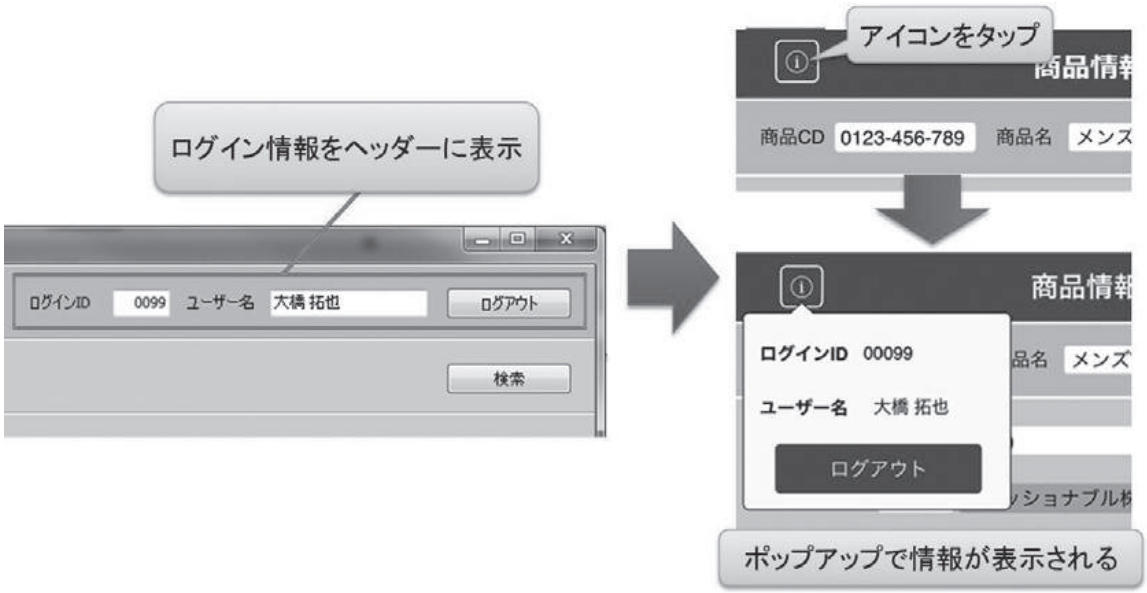


図20



ソース1

```
{*************************************************************************
 目的：ログイン情報押下時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.Button2Click(Sender: TObject);
begin
  // ポップアップ表示切替
  Popup1.Visible := not Popup1.Visible;
  Popup1.BringToFront;
end;

{*************************************************************************
 目的：画面タッチ時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.FormTouch(Sender: TObject; const Touches: TTouches;
  const Action: TTouchAction);
begin
  // ポップアップ終了
  if Popup1.Visible and
     (not Popup1.PointInObject(Touches[0].Location.X, Touches[0].Location.Y)) and
     (not Button2.PointInObject(Touches[0].Location.X, Touches[0].Location.Y)) then
  begin
    Popup1.Visible := False;
  end;
end;
```

アイコン以外の場所にタッチした場合、
ポップアップを非表示にする

図21

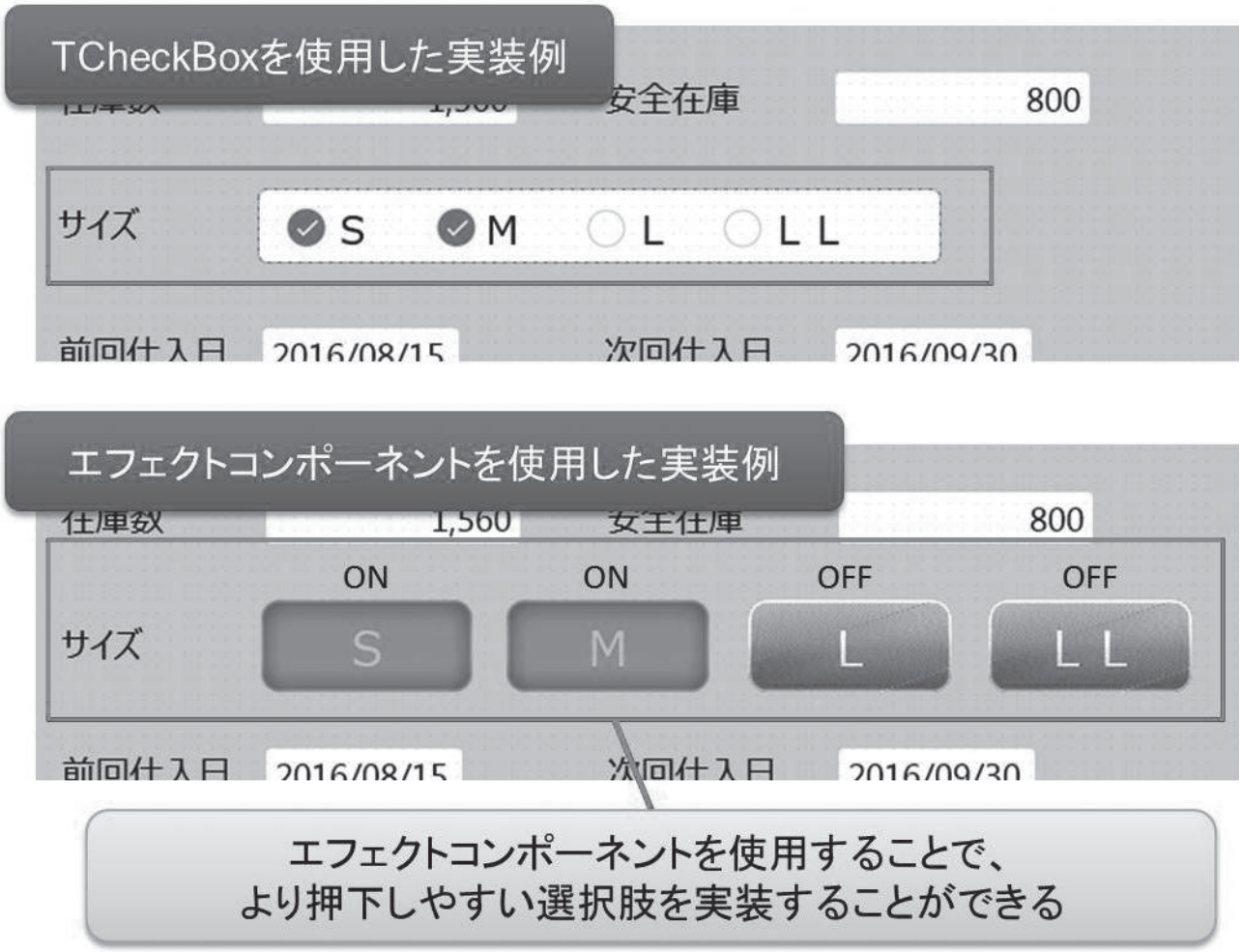

TCheckBoxを使用した実装例
安全在庫
800
サイズ
S
M
L
L L
前回仕入日
2016/08/15
次回仕入日
2016/09/30
エフェクトコンポーネントを使用した実装例
在庫数
1,560
安全在庫
800
ON
ON
OFF
OFF
サイズ
S
M
L
L L
前回仕入日
2016/08/15
次回仕入日
2016/09/30
エフェクトコンポーネントを使用することで、
より押下しやすい選択肢を実装することができる

図22

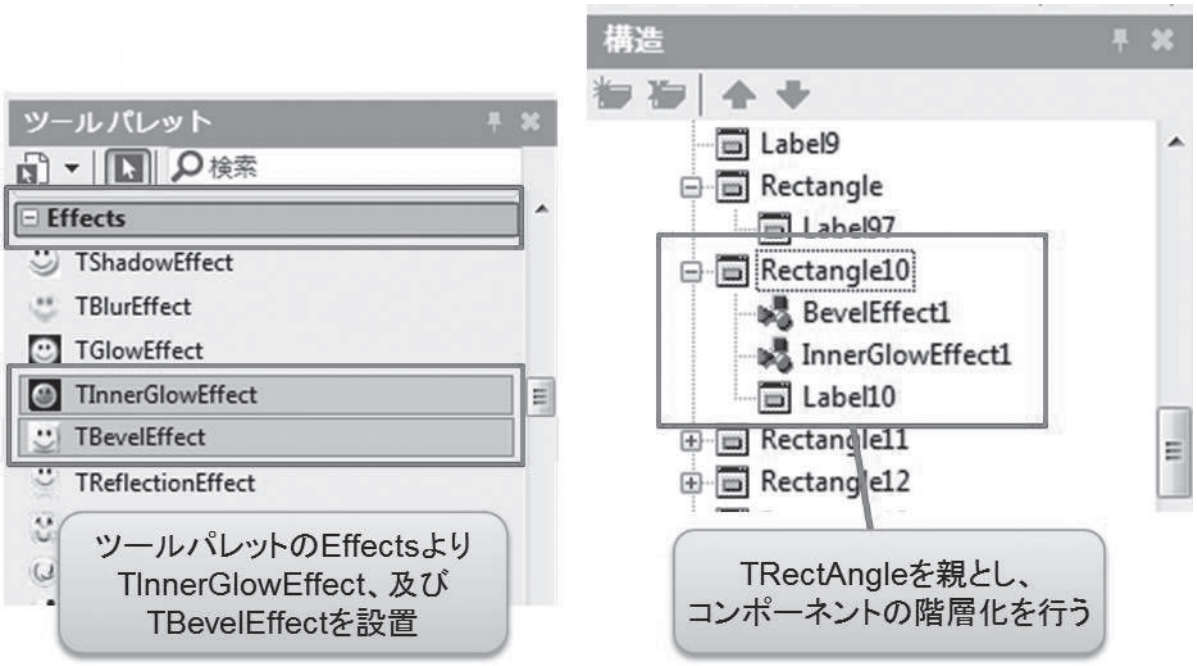
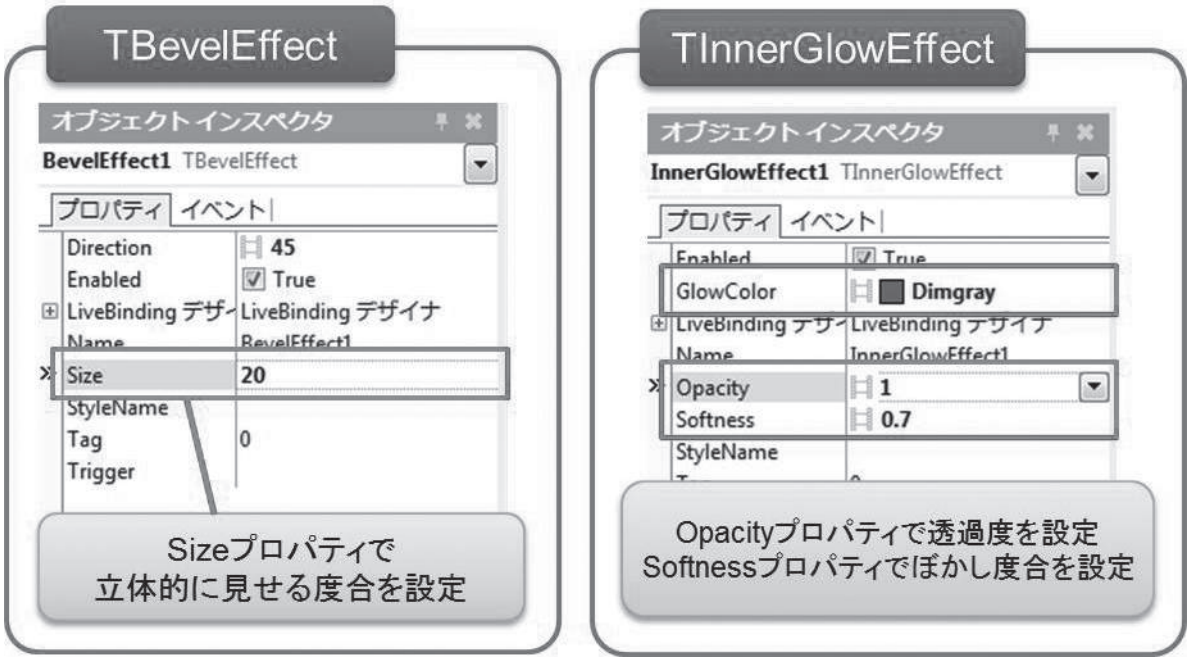

ツールパレット
検索
Effects
TShadowEffect
TBlurEffect
TGlowEffect
TInnerGlowEffect
TBevelEffect
TReflectionEffect
ツールパレットのEffectsより
TInnerGlowEffect、及び
TBevelEffectを設置
構造
Label9
Rectangle
Label97
Rectangle10
BevelEffect1
InnerGlowEffect1
Label10
Rectangle11
Rectangle12
TRectAngleを親とし、
コンポーネントの階層化を行う

図23



ソース2

```
  private
    { private 宣言 }
    FPushFlg: Boolean;  // 押下フラグ
  public
    { public 宣言 }
  end;

var
  FireMonkeyForm: TFireMonkeyForm;

implementation

{$R *.fmx}

{*******************************************************************************
 目的: 選択ボタン 押下時処理
 引数:
 戻値:
*******************************************************************************}
procedure TFireMonkeyForm.Label1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Single);
begin
  if FPushFlg then
  begin
    BevelEffect1.Enabled      := True;
    InnerGlowEffect1.Enabled  := False;
    Label1.Opacity            := 1;
    FPushFlg                  := False;
  end
  else
  begin
    BevelEffect1.Enabled      := False;
    InnerGlowEffect1.Enabled  := True;
    Label1.Opacity            := 0.5;
    FPushFlg                  := True;
  end;
end;
```

選択状態ON/OFF 切替用フラグ

ON から OFF へ変更の処理

OFF から ON へ変更の処理

【処理内容】
・TBevelEffectの切替(Enabled)
・TInnerGlowEffectの切替(Enabled)
・TLabelの透明度設定(Opacity)
・フラグの切替